UAS See-and-Avoid Strategy using a Fuzzy Logic Controller Optimized by Cross-Entropy in Scaling Factors and Membership Functions

Changhong Fu , Miguel A. Olivares-Mendez , Pascual Campoy , Ramon Suarez-Fernandez

I. INTRODUCTION

The autonomous localization for Unmanned Aerial System (UAS) has been researched and developed fruitfully in the robot community recently. However, Global Positioning System (GPS)-based flight [1] can not fly in the indoor environments where there is no GPS service. Laser range finder-based flying [2] can not work in the open and broad places because of its limited detection distance. And Motion Capture System-based flight [3] just work in the indoor and local space where many expensive cameras with high speed are constructed. Considering the cost, size, power consumption, weight and surrounding information different sensors can obtained, camera is the best onboard option. It can achieve the Visual Odometry (VO) [4] to estimate the 6D pose of UAS. Monocular and stereo methods as the main camera-based approach are widely used in UAS, especially for quadrotor helicopter. But the stereo camera has its limitation when the baseline is much smaller than the distance between UAS with the target, and its cost, weight and power comsumption are higher compared to monocular camera. Hence, with the improvement of performance/price in camera and development of UAS platform, e.g. AscTec Products [5] and AR.Drone Parrot [6], many works in Autonomous Systems Lab [7] and Computer Vision Group [8] have been researched based on the advanced monocular Simultaneous Localization and Mapping (SLAM) techniques, they also overcomed the drawback of monocular SLAM to estimate the real absolute scale to environments by fusing air pressure sensor, IMU, ultrasonic sensor in order to navigate the UAS accurately.

Sense-and-avoid (SAA) problem has been identified as one of the most significant challenges facing the integration of aircraft into the airspace. Here, the term "sense" relates to the use of sensor information to automatically detect possible aircraft conflicts, whilst the term "avoid" relates to the automated control actions used to avoid any detected collisions [9]. The onboard single or multiple sensors can provide the sense-and-avoid capability for flying aircraft. However, as what has been mentioned above, the camera sensor is the best onboard candidate for collision avoidance in UAS. There are many works about sense-and-avoid applications using computer vision algorithms. He et al [10] present a vision-based obstacle avoidance method using motion field information. Vision-based 3D geometry estimation for static obstacles has been proposed in [11] to control the UAS flying in the urban environments. Some works, such as [12] have developed the optical flow-based approach to obtain the image depth to avoid the obstacles for UAS. The unique light field in the real-time images is used to extract the horizon lines within the sea-sky, soil-sky and forest-sky in [13] to avoid collision in sky. And some researches, e.g. [14], have presented how to detect and track the point-like UAS with the far distance only using the visual information.

Many classical controllers have been researched and developed for UAS in the past decades. However, the uncertainty, inaccuracy, approximation and incompleteness problems widely exist in real controlling techniques. Luckily, these issues can be well dealt with the Soft Computing (SC) approaches. Fuzzy Logic Controller (FLC) is one of the most active and fruitful SC method. This technique is based on the fuzzy logic that imitates human thinking and decision making with natural language. Its essential part is a set of linguistic control rules related by the dual concepts of fuzzy implication and the compositional rule of inference. In other words, FLC provides an algorithm which can convert the linguistic control strategy based on expert knowledge into an automatic control strategy. Experience

shows that the FLC yields results superior to those obtained by conventional control algorithms. Several recent works have proved the advantages of Fuzzy Logic Controller and its optimization. Especially, the literatures about optimized FLC for UAS are: a robustness comparison between model-based with self-tunable fuzzy inference system (STFIS) has been studied to control a drone in presence of disturbances [15]. The classical and multi-objective genetic algorithm (GA) based fuzzy-genetic autopilot are also designed and used for UAS [16], in that work the benefits for the time response characteristics, the robustness and the adaptation of fuzzy controller with respect to the large commands were validated. An adaptive neuro-fuzzy inference system (ANFIS) based controller for UAS [17] was developed to adjust its altitude, the heading and the speed together.

The Cross-Entropy method derives its name from the Cross-Entropy (or Kullback-Leibler) distance, which is a fundamental concept of modern information theory. The method was motivated by an adaptive algorithm for estimating probabilities of rare events in complex stochastic networks, which involves variance minimization. In a nutshell, the CE method involves an iterative procedure where each iteration can be broken down into two phases. In the first stage, a random data sample (e.g. scaling factors or a set of membership functions for Fuzzy Logic Controller) is generated according to a specified mechanism. Then, the parameters of the random mechanism are updated based on the data in order to produce a *"better"* sample in the next iteration. The CE method provides a unifying approach to simulation and optimization [18]. Several applications demonstrated the power of the CE method, such as ship detection, CT image reconstruction, blind multiuser acquisition, power system reliability evaluation, optimal path planning, antena selection, and motion planning. This optimization method was also used to tune controllers in only three literatures, but CE is just used to optimize the scaling factors in different controllers. Bodur [19] has applied the CE to optimize the scaling factors for PID controller in a simulated inverted pendulum. Haber et al. [20] also use CE to tune the scaling factors for a Fuzzy PD controller in cutting force regulation of a drilling process. And our previous work [21] presented a CE-based optimization for scaling factors in a PID Fuzzy controller to command an high dynamic aerial vehicle for avoiding a static obstacle with special color.

The outline of the paper is as follows. The Problem Statement is described in Section II. We introduced the UAMVIS in Section III. In Section IV, we designed the fuzzy controller with its initial scaling factors, membership functions and rule base. Then, the Cross-Entropy theory and its optimization method for fuzzy controller are introduced. In Section VI, the simulation stages and results are shown. In Section VII, the real flight results has been given and discussed. Finally, the concluding remarks and future work are presented in the Section VIII.

II. Problem Statement

Many typical civil tasks, such as wildfire monitoring in tree lines, disaster rescue in mountains and fault inspection for buildings or bridges in cluttered urban, are carrying out by UAS currently, and a field study after Hurricane Katrina [22] concluded one of the most important recommendations for autonomy UAS is that the minimum emergent standoff distance from inspected structures is 2-5m. This paper aims to discuss how to prevent crashes by UAS itself when UAS fly into this recommended distance based on the former working distance. However, few works have addressed this problem appropriately, e.g. providing real-time full 6D pose information for UAS, the main reasons for UAS, especially for quadcopter, depend on the capabilities and control approaches. As what we has discussed in the Section I, UAS see-and-avoid task can benefit from onboard camera directly. Considering a flying quadcopter, e.g. AR. Drone Parrot, moving forwardly with a constant flight speed to an obstacle, e.g. wall, where, the heading of quadrotor helicopter is parallel to the normal vector of the obstacle. The control goal is to command it to avoid the obstacle, at least making it flying parallelly to the obstacle with a safe distance. Figure 1 shows the collision avoidance strategy, we divided the whole area into three parts, and the Dangerous Alarm Area (DAA) is set based on our quadcopter size (52.5x51.5cm) and its inertance, as shown in the Figure 2, this area is 1 meter in length, Safe Avoiding Area (SAA) is designed based on the recommended distance from 1 meter to 4 meter in length, and the left area is Normal Fly Area (NFA). The Start Point (S) can be set on any place in the NFA. For this problem, different constant flight speeds and sizes of SAA will be tested in the simulations and real flights.

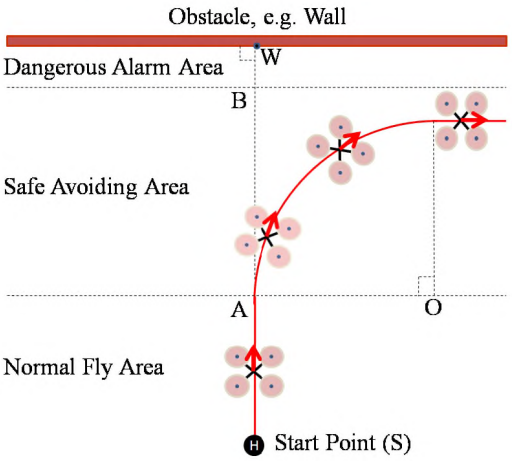


Fig. 1: **2D** Description for See-and-Avoid Task.

III. Unmanned Aerial Monocular Vision-IMU System

Unmanned Aerial Monocular Vision-IMU System (UAMVIS) is a kind of advanced UAS locating by itself with single monocular camera and IMU sensor. Here, for monocular vision, several visual odometry and visual SLAM frameworks have been launched in recent years. However, the keyframe-based Parallel Tracking and Mapping (PTAM) [23] was used for UAS becasue of its parallel processing threads, fast response performance and widely application in common scenarios. It can provide 6 Degrees of Freedom

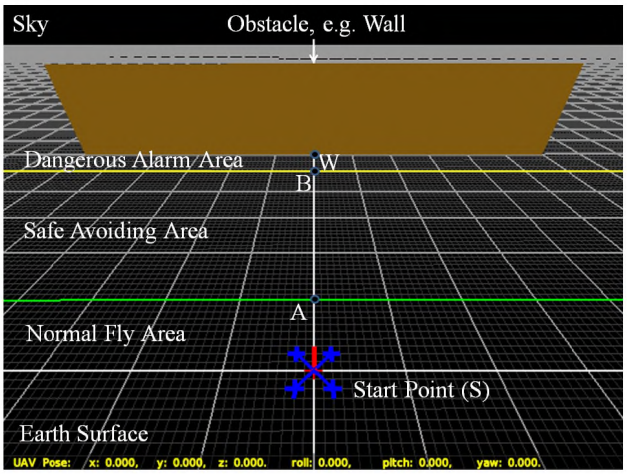
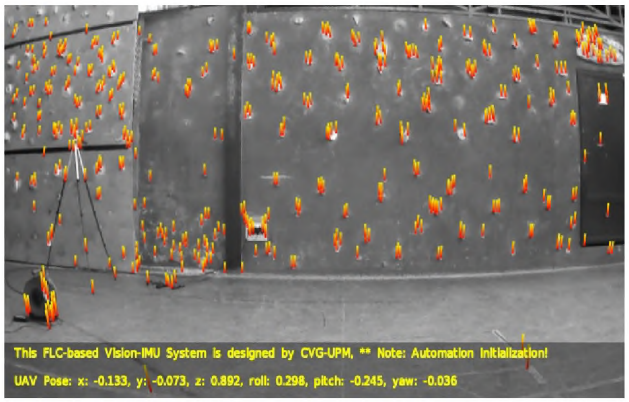
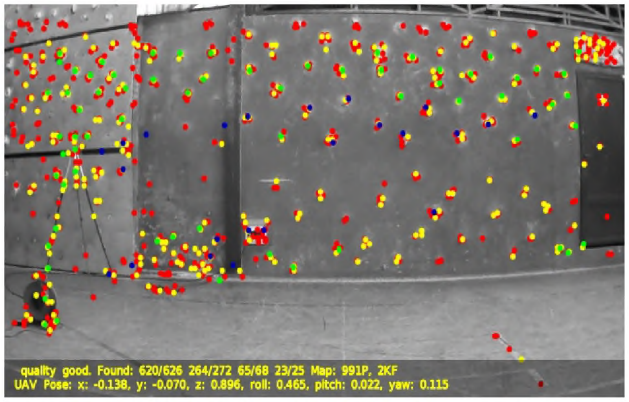


Fig. 2: Real-time **3D** Synchronization Map, where, the scale of each big grid in white is equal to 1 meter in reality.



(a) FLC-based UAMVIS during Automation Initilization Stage, where, the orange-yellow line stands for the tracked keypoint (FAST corners) movement from the first keyframe to current frame.



(b) FLC-based UAMVIS during Visual Fuzzy Servoing Stage, where, the dot represents the localization of keypoint. And the colors correspond to which pyramid-level the keypoint is measured in.

Fig. 3: Real-time Processing Images of FLC-based UAMVIS.

(DOF) estimation. For Inertial Measurement Unit (IMU), it is a 3D acceleration and rotation estimator. Jakob Engel et al [24] has presented an approach to enable the AR. Drone Parrot to accurately fly various figures, which has been published as the open source in the Robot Operating System (ROS) [25] and includes three main components: a monocular SLAM system, an extended kalman filter for data fusion and state estimation and a PID controller. Considering the precise 6D estimation in his works, we developed a Fuzzy Logic Controller-based UAMVIS using his first two modules. Figure 3a and 3b show the automation initialization of PTAM and real-time processing image fusing visual pose estimation with IMU measurement.

IV. Fuzzy Logic Controller

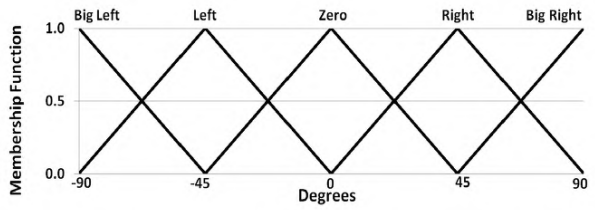
Considering the non-linearity of the system and high dynamic environment, a fuzzy controller was designed to control the orientation of the aircraft. The huge experience also proved that Fuzzy Logic-based Soft-computing technique can provide better performances in controllers. As previously developed controllers in [26], this controller also was developed using the MOFS (Miguel Olivares' Fuzzy Software).

The controller is a PID like controller, so it has three inputs. The first one is the estimated angle error in degrees between the reference with aircraft heading. Other inputs are the derivate value and the integral on time of this angle error estimation. The output is a command in degrees per seconds to change the heading of the aircraft. The initial scaling factors without CE optimization has a default value equal to one. Since the avoiding task is identical for right or left side avoiding, and the aircraft has a symmetric design with the same behavior for left and right heading movements, the heading FLC has a symmetric definition of the inputs, output and the rule base.

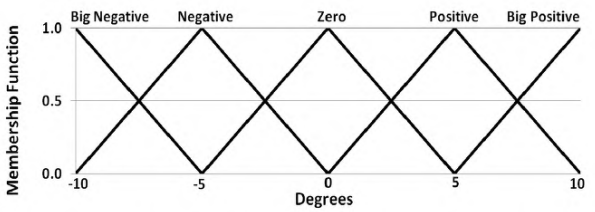
Figure 4 shows the initial definition of the inputs and output. Each input has 5 sets and the output has 9 sets. The symmetry of the FLC implies that any modification of the left side of each variable (input and output) can be applied to the right side.

The rule base was designed using the heuristic information based on expert knowledge. Each rule without CE optimization has a default weight equal to one. In that way, each rule has the same importance and affects to the FLC behavior in the same way. The three inputs of the controller imply that the rule base has a cube disposition of $5 \times 5 \times 5$. Here, 5 tables of 5×5 are presented in order to show the rule base. Each one is related to one of the 5 linguistic values for the third variable: the integral of error. Table I shows the rule base slide for the *zero* value of the error integral on time. Table II shows the slide for the *negative* value. Table III shows the slide for the *big negative* value. Table IV shows the slide for the *positive* value. Finally Table V shows the slide for the *big positive* value.

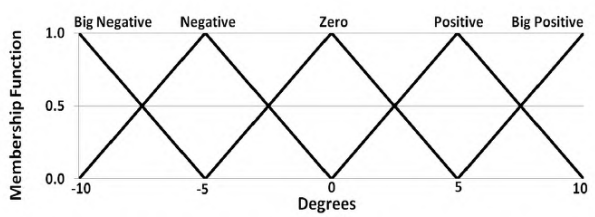
The product t-norm is used for rules conjunction, and the defuzzification method used in this approach is a modification of the Height Weight method. It introduce the value of the weight assigned to each rule in the defuzzification process. Equation 1 shows the defuzzification method.



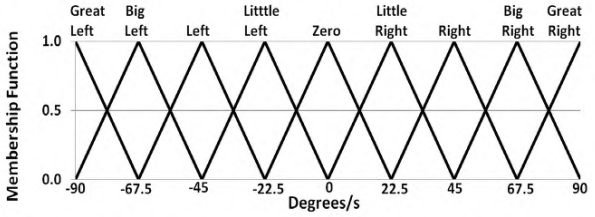
(a) **First input** membership functions. Yaw Error, **without** CE optimization



(b) **Second input** membership functions, Derivative of Yaw Error, **without** CE optimization



(c) **Third input** membership functions, Integral of Yaw Error, **without** CE optimization



(d) **Output** membership functions, Yaw Command, **without** CE optimization

Fig. 4: The Initial Definition for Membership Functions of the Fuzzy Logic Controller **before** CE Optimization.

TABLE I: Rules based on the third input (integral of the error) equal to **Zero**, **before** CE Optimization

Dot error/error	Big Left	Left	Zero	Right	Big Right
Big Negative	Great Left	Big Left	Left	Little Left	Zero
Negative	Big Left	Left	Little Left	Zero	Little Right
Zero	Left	Little Left	Zero	Little Right	Right
Positive	Little Left	Zero	Little Right	Right	Big Right
Big Positive	Zero	Little Right	Right	Big Right	Great Right

TABLE II: Rules based on the third input (integral of the error) equal to **Negative**, **before** CE Optimization

Dot error/error	Big Left	Left	Zero	Right	Big Right
Big Negative	Big Left	Left	Little Left	Zero	Little Right
Negative	Left	Little Left	Zero	Little Right	Right
Zero	Little Left	Zero	Little Right	Right	Big Right
Positive	Zero	Little Right	Right	Big Right	Great Right
Big Positive	Little Right	Right	Big Right	Great Right	Great Right

TABLE III: Rules based on the third input (integral of the error) equal to **Big Negative**, **before** CE Optimization

Dot error/error	Big Left	Left	Zero	Right	Big Right
Big Negative	Left	Little Left	Zero	Little Right	Right
Negative	Little Left	Zero	Little Right	Right	Big Right
Zero	Zero	Little Right	Right	Big Right	Great Right
Positive	Little Right	Right	Big Right	Great Right	Great Right
Big Positive	Right	Big Right	Great Right	Great Right	Great Right

TABLE IV: Rules based on the third input (integral of the error) equal to **Positive**, **before** CE Optimization

Dot error/error	Big Left	Left	Zero	Right	Big Right
Big Negative	Great Left	Great Left	Big Left	Left	Little Left
Negative	Great Left	Big Left	Left	Little Left	Zero
Zero	Big Left	Left	Little Left	Zero	Little Right
Positive	Left	Little Left	Zero	Little Right	Right
Big Positive	Little Left	Zero	Little Right	Right	Big Right

TABLE V: Rules based on the third input (integral of the error) equal to **Big Positive**, **before** CE Optimization

Dot error/error	Big Left	Left	Zero	Right	Big Right
Big Negative	Great Left	Great Left	Great Left	Big Left	Left
Negative	Great Left	Great Left	Big Left	Left	Little Left
Zero	Great Left	Big Left	Left	Little Left	Zero
Positive	Big Left	Left	Little Left	Zero	Little Right
Big Positive	Left	Little Left	Zero	Little Right	Right

$$y = \frac{\sum_{l=1}^{M} \overline{y}^l \prod_{i=1}^{N} \left(\mu_{x_i^l}(x_i) w_i\right)}{\sum_{l=1}^{M} \prod_{i=1}^{N} \left(\mu_{x_i^l}(x_i) w_i\right)} \quad (1)$$

Where N and M represent the number of inputs variables and total number of rules, respectively. $\mu_{x_i^l}$ denotes the merbership function of the lth rule for the ith input variable. $\overline{y}^l$ represents the output of the lth rule. w_i corresponds to the weight of the ith rule that could takes values from 0 to 1.

V. Cross Entropy Optimization

The Cross-Entropy (CE) method is a new approach in stochastic optimization and simulation. It was developed as an efficient method for the estimation of rare-event probabilities. The CE method has been successfully applied to a number of difficult combinatorial optimization problems. An application of this method was presented for optimization of the scaling factors and membership functions of a Fuzzy controller. Next, the method and the Fuzzy controller optimization approach are proposed. A deeper explanation of the Cross-Entropy method for general uses is presented on [18].

A. Optimization Principle

The CE method is iterative and based on the generation of a random data sample $(x_1, ..., x_N)$ in the χ space according to a specified random mechanism. A reasonable option is to use a probability density function (pdf) such as the normal distribution. Let $g(-, v)$ be a family of probability density functions in χ parameterized by a real value vector $v \in \Re$: $g(x, v)$. Let ϕ be a real function on χ, so the aim of the CE method is to find the minimum (like in our case) or maximum of ϕ over χ, and the corresponding states x^* satisfying this minimum/maximum: $\gamma^* = \phi(x^*) = min_{x \in \chi} \phi(x)$.

In each iteration the CE method generates a sequence of $(x_1, ..., x_N)$ and $\gamma_1 ... \gamma_N$ levels such that γ converges to γ^* and x to x^*. We are concerned with estimating the probability $l(\gamma)$ of an event $E_v = \{x \in \chi \mid \phi(x) \geq \gamma\}, \gamma \in \Re$.

Defining a collection of functions for $x \in \chi, \gamma \in \Re$.

$$I_v(x, \gamma) = I_{\{\chi(x_i) > \gamma\}} = \begin{cases} 1 & if \phi(x) \leq \gamma \\ 0 & if \phi(x) > \gamma \end{cases} \quad (2)$$

$$l(\gamma) = P_v(\chi(x) \geq \gamma) = E_v \cdot I_v(x, v) \quad (3)$$

where E_v denotes the corresponding expectation operator.

In this manner, Equation 3 transforms the optimization problem into an stochastic problem with very small probability. The variance minimization technique of importance sampling is used, in which the random sample is generated based on a pdf h. Being the sample $x_1, ..., x_N$ from an importance sampling density h on ϕ and evaluated by:

$$\hat{l} = \frac{1}{N} \cdot \sum_{i=1}^{N} I_{\{\chi(x_i) > \gamma\}} \cdot W(x_i) \quad (4)$$

Where $\hat{l}$ is the importance sampling and $W(x) = \frac{g(x,v)}{l}$ is the likelihood ratio. The search for the sampling density $h^*(x)$ is not an easy task because the estimation of $h^*(x)$ requires that l be known $h^*(x) = I_{\{\chi(x_i) > \gamma\}} \cdot \frac{g(x,v)}{l}$. So the referenced parameter v^*, must be selected such the distance between h^* and $g(x, v)$ is minimal, thereby the problem is reduced to a scalar case. A way to measure the distance between two densities is the Kullback-Leibler, also known as Cross-Entropy:

$$D(g, h) = \int g(x) \cdot ln\ g(x) dx - \int g(x) \cdot ln\ h(x) dx \quad (5)$$

The minimization of $D(g(x, v), h^*)$ is equivalent to maximize $\int h^* ln[g(x, v)] dx$ which implies that $max_v D(v) = max_v E_p \left(I_{\{\chi(x_i) > \gamma\}} \cdot ln\ g(x, v)\right)$, in terms of importance sampling it can be rewritten as:

$$max_v \hat{D}(v) = max \frac{1}{N} \sum_{i=1}^{N} I_{\{\chi(x_i) > \gamma\}} \cdot \frac{p_x(x)}{h(x_i)} \cdot ln\ g(x_i, v) \quad (6)$$

Note that h is still unknown, therefore the CE algorithm will try to overcome this problem by constructing an adaptive sequence of the parameters $(\gamma_t \mid t \geq 1)$ and $(v_t \mid t \geq 1)$.

B. FLC Optimization Description

The Cross-Entropy optimization method was used, in this work, to optimize two different parts of the Fuzzy controller. Following the process of manual to tune or optimize Fuzzy controllers defined by Zheng in 1992 [27], here a Macroscopic and Medium-size optimization are presented. As is recommended in that work the two optimization were done independently. Firstly, the Macroscopic optimization is done to improve the behavior of the controller. Once the above optimization process is finished the Medium-size optimization was executed, modifying the membership functions of each variable. In this case Cross-Entropy method was applied to modify the position and size of the sets of the membership functions. In the second optimization phases some sets of membership functions are nearly overlapped between each other, so it is possible to delete some sets.

The algorithm used for the optimization is almost the same with differences in dimensionality size of the problem to optimize. The CE method generates a set of N fuzzy controllers $x_i = (x_{i1}, x_{i2}, ..., x_{ih})$ with $g(x, v) = (g(x_1, v), g(x_2, v), ..., g(x_h, v))$ and calculates the objective function value for each controller. The controllers parameters $x_{i1}, x_{i2}, ..., x_{ih}$ correspond to scaling factors in the first optimization and then the position of the membership functions sets in the second one. After all the generated controllers are simulated an update of $g(x, v)$ is done using a set of best controllers. The number of best controllers used to update the pdf is denoted by N^{elite}. Then a new set of controllers is generated to be tested. The process finish when the maximum number of iteration are reached. A generic version of the optimization process for fuzzy controllers is presented in the Algorithm 1.

Algorithm 1 Cross-Entropy Algorithm for Fuzzy controller optimization

1. Initialize $t = 0$ and $v(t) = v(0)$
2 Generate a sample of N controllers: $(x_i(t))_{1 \leq i \leq N})$ from $g(x, v(t))$, being each $x_i = (x_{i1}, x_{i2}, ..., x_i h)$
3. Compute $\phi(x_i(t))$ and order $\phi_1, \phi_2, ..., \phi_N$ from smallest ($j = 1$) to biggest ($j = N$).
Get the N^{elite} first controllers $\gamma(t) = \chi_{[N^{elite}]}$.
4. Update $v(t)$ with
$v(t+1) = arg_v min \frac{1}{N^{elite}} \sum_{j=1}^{N^{elite}} I_{\{\chi(x_i(t)) \geq \gamma(t)\}} \cdot ln\, g(x_j(t), v(t))$
5. Repeat from step 2 until convergence or ending criterion.
6. Assume that convergence is reached at $t = t^*$, an optimal value for ϕ can be obtained from $g(., v(t)^*)$.

For both optimization process a Normal (Gaussian) distribution function was used. The mean μ and the variance σ of each h parameters are calculate for each t iteration as $\tilde{\mu}_{th} = \sum_{j=1}^{N^{elite}} \frac{x_{jh}}{N^{elite}}$ and $\tilde{\sigma}_{th} = \sum_{j=1}^{N^{elite}} \frac{(x_{jh} - \mu_{jh})^2}{N^{elite}}$. The mean vector $\tilde{\mu}$ should converge to γ^* and the standard deviation $\tilde{\sigma}$ to zero.

In order to obtain a smooth update of the mean and the variance we use a set of parameters (β, α, η), where α is a constant value used for the mean, η is a variable value which is applied to the variance to avert the occurrences of $0s$ and $1s$ in the parameter vectors, and β is a constant value which modify the value of $\eta(t)$.

$$\begin{aligned} \eta(t) &= \beta - \beta \cdot (1 - \tfrac{1}{t})^q \\ \hat{\mu}(t) &= \alpha \cdot \tilde{\mu}(t) + (1 - \alpha) \cdot \hat{\mu}(t-1) \\ \hat{\sigma}(t) &= \eta(t) \cdot \tilde{\sigma}(t) + (1 - \eta(t)) \cdot \hat{\sigma}(t-1) \end{aligned} \tag{7}$$

Where $\hat{\mu}(t-1)$ and $\hat{\sigma}(t-1)$ are the previous values of $\hat{\mu}(t)$ and $\hat{\sigma}(t)$. The values of the smoothing update parameters are $0.4 \leq \alpha \leq 0.9$, $0.6 \leq \beta \leq 0.9$ and $2 \leq q \leq 7$. In order to get an optimized controller, the objective function named Integral Time of Square Error (ITSE) is chosen.

VI. Training Stages and Results

This section presents the results of SFs and MFs optimization in simulation collision avoidance task. The training environment has been constructed strictly as described in Section II, and a set of simulink blocks were implemented, including Quadrotor model, virtual camera, obstacle, fuzzy controller and flowchart used to manage the tests. In each iteration, different constant pitch speeds were sent to test all the controllers, and roll speed was always zero. The initial parameters for Cross-Entropy were set based on [20], [21] and [28]. Figure 5 shows the whole training process.

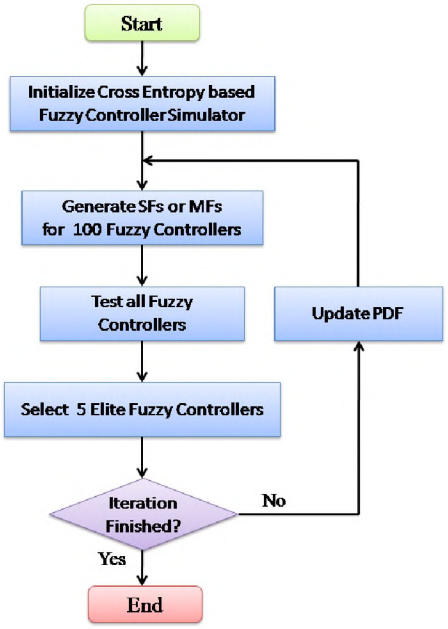


Fig. 5: Flowchart of Cross-Entropy Optimization for Fuzzy Controller in Scaling Factors and Membership Functions.

A. Scaling Factors Optimization Results

Figure 6 shows the control loop during the scaling factors (K_p, K_d, K_i) optimization stage. The evolution of the value to optimize could be shown with mean and sigma associated of each scaling factors. Both values can be used to represent the Probability Density Function (PDF) in each iteration. Figure 7 shows the evolution of the PDF for the scaling factors of first input in Fuzzy Controller. Similarily, the final optimal scaling factors for second and third input are 0.03 and -0.5003 in 100 iterations, respectively.

B. Membership Functions Optimization Results

After obtained the optimal Scaling Factors (K_{po}, K_{do}, K_{io}) for fuzzy controller, the Membership Functions should be optimized. Figure 8 shows the control loop during the membership functions optimization stage. Considering the membership functions are symmetric, any position modification of the left side of each variable (input and output) can be applied to the right side. Figure 9 shows the evolution of the PDF for the membership functions of first input (Left) of Fuzzy Controller. Similarily, the final optimal membership function for second input (Negative), third input (Negative), Big Left, Left and Little Left in output are -8.1166, -9.9782, -88.974, -88.191 and -74.952

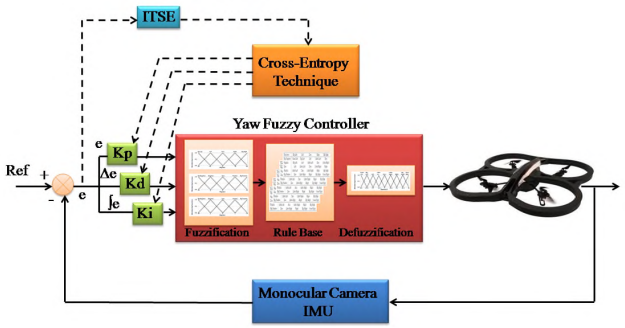


Fig. 6: Cross-Entropy Optimization for Scaling Factors in Fuzzy Controller.

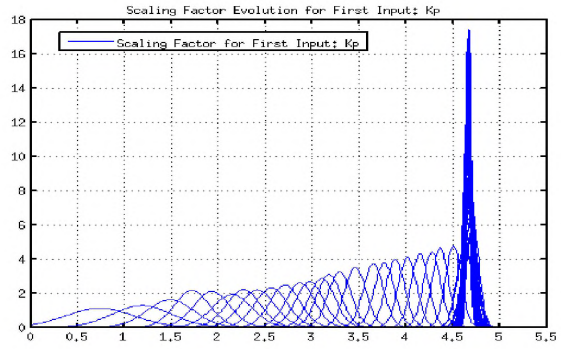


Fig. 7: Cross-Entropy Optimization based Evolution of the probability density function for the Scaling Factor of First Input in Fuzzy Controller (from Left to Right). The Final Optimal Scaling Factor for First Input is 4.6739 in 100 iterations.

in 100 iterations, respectively. Hence, the final optimal membership function for second input (Positive), third input (Positive), Big Right, Right and Little Right in output are 8.1166, 9.9782, 88.974, 88.191 and 74.952 in 100 iterations, respectively, which also have been shown in Figure 10.

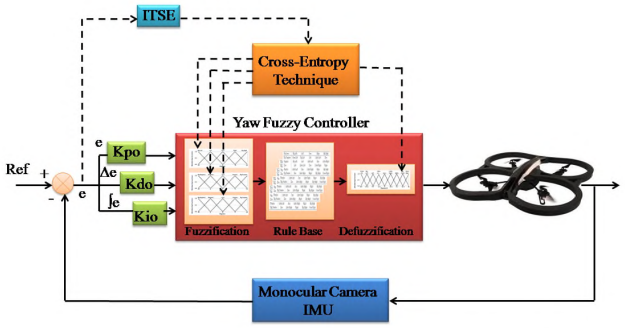


Fig. 8: Cross-Entropy Optimization for Membership Functions based on the Optimized Scaling Factors in Fuzzy Controller.

Figure 10 shows the optimized membership functions,

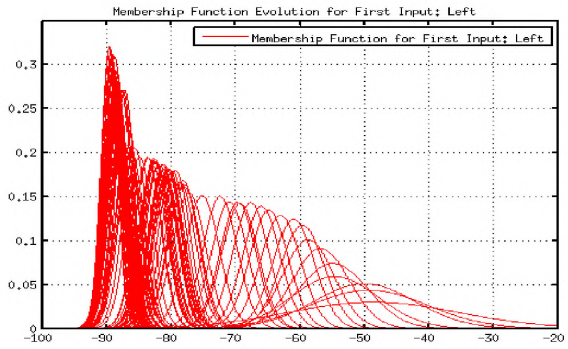


Fig. 9: Cross-Entropy Optimization based Evolution of the probability density function for the Membership Function of First Input (Left) in Fuzzy Controller (from Right to Left). The Optimal Membership Function for Left is -89.6960 and Right is 89.6960 according to the Symmetry of MFs in 100 iterations.

where, two sets of membership functions has been reduced in Figure 10a and 10c, and four sets in Figure 10d. These reductions lead to the cancellation of rule bases directly. Table VI, VII and VIII shows the final rule bases, 64% of rules has been cancelled from 125 rules to 45 rules, where, Table III (25 rules) and V (25 rules) have been cancelled, 10 rules have been reduced in Table I, II and IV, respectively.

TABLE VI: Rules for the third input (integral of the error) equal to **Zero**, **after** CE Optimization

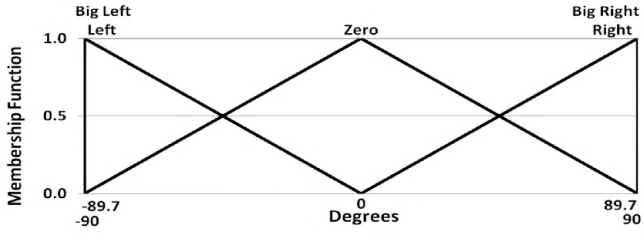
Dot error/error	Left	Zero	Right
Big Negative	Left	Left	Little Left
Negative	Left	Little Left	Zero
Zero	Little Left	Zero	Little Right
Positive	Zero	Little Right	Right
Big Positive	Little Right	Right	Right

TABLE VII: Rules for the third input (integral of the error) equal to **Negative**, **after** CE Optimization

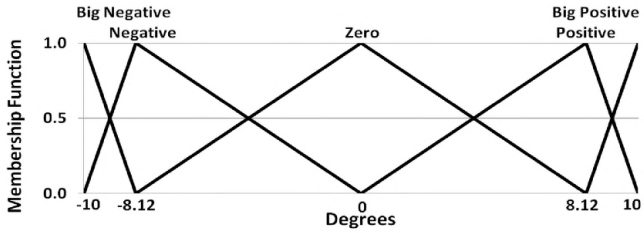
Dot error/error	Left	Zero	Right
Big Negative	Left	Little Left	Zero
Negative	Little Left	Zero	Little Right
Zero	Zero	Little Right	Right
Positive	Little Right	Right	Right
Big Positive	Right	Right	Right

TABLE VIII: Rules for the third input (integral of the error) equal to **Positive**, **after** CE Optimization

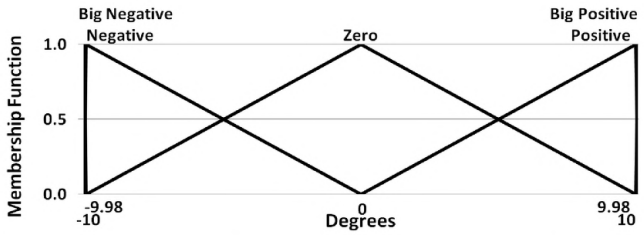
Dot error/error	Left	Zero	Right
Big Negative	Left	Left	Left
Negative	Left	Left	Little Left
Zero	Left	Little Left	Zero
Positive	Little Left	Zero	Little Right
Big Positive	Zero	Little Right	Right



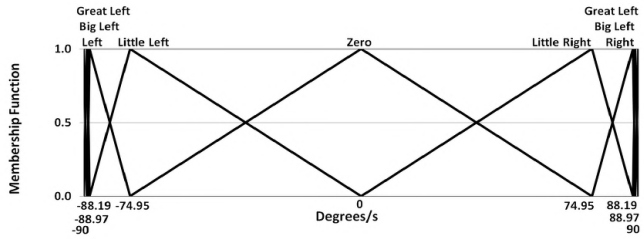
(a) **First input** membership functions, Yaw Error, **after** CE optimization, where, the **Left (Right)** has been optimized to -89.6960 (89.6960) compared to the Figure 4(a).



(b) **Second input** membership functions, Derivative of Yaw Error, **after** CE optimization, where, the **Negative (Positive)** has been optimized to -8.1166 (8.1166) compared to the Figure 4(b).



(c) **Third input** membership functions, Integral of Yaw Error, **after** CE optimization, where, the **Negative (Positive)** has been optimized to -9.9782 (9.9782) compared to the Figure 4(c).

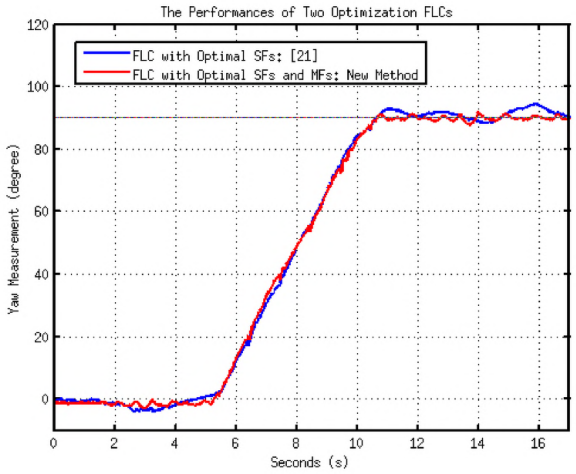


(d) **Output** membership functions, Yaw Command, **after** CE optimization, where, the **Big Left, Left, Little Left (Big Right, Right, Little Right)** has been optimized to -88.974, -88.191, -74.952 (88.974, 88.191, 74.952) compared to the Figure 4(d).

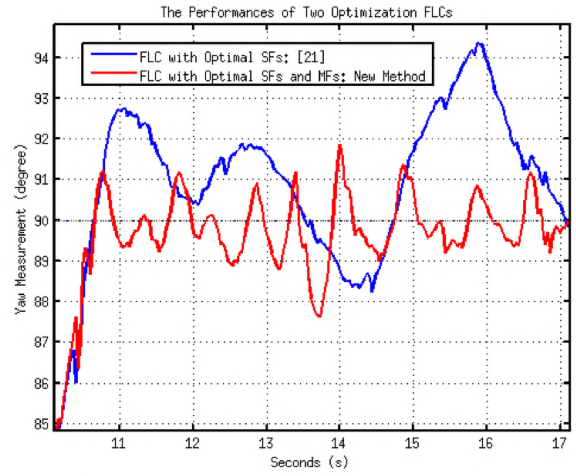
Fig. 10: The Final Optimized Results of Membership Functions of the Fuzzy Logic Controller.

VII. Real Flights and Discussion

After obtained the optimal SFs and MFs, a large number of real tests have been done with a quadcopter, AR.Drone Parrot [6], in order to compare the performances using different optimized fuzzy controllers, pitch speeds and sizes of SAA.



(a) Measurements of Heading Angle in the Whole Task.



(b) Enlarged Image for Steady State Performances.

Fig. 11: The Performances of Two Optimized Fuzzy Controller. Once the quadrotor take off and finish initialization stage for localization, it flies one meter towards the obstacle in NFA. Then the reference command (90 degree) is sent by itself in SAA, and visual fuzzy servoing is activated to avoid collision.

Figure 11 shows the performances of two optimized fuzzy controllers in collision avoidance task, flight speed is 0.2m/s, SAA in length is 2 meters, and the start point is 4 meters from the obstacle. In this case, both controllers avoided the obstacle successfully and did not fly into the dangerous alarm area. Figure 11a shows the measurement of yaw angle in the whole see-and-avoid task, and figure 11b is the enlarged image to show the performance in steady state, where the average RMSE is 4.121 degree for SFs optimized fuzzy controller, while the average RMSE is 1.921 degree for SFs and MFs optimized fuzzy controller in all the tests.

Figure 12 shows the external images and real-time processing images of different states in avoiding collision task.

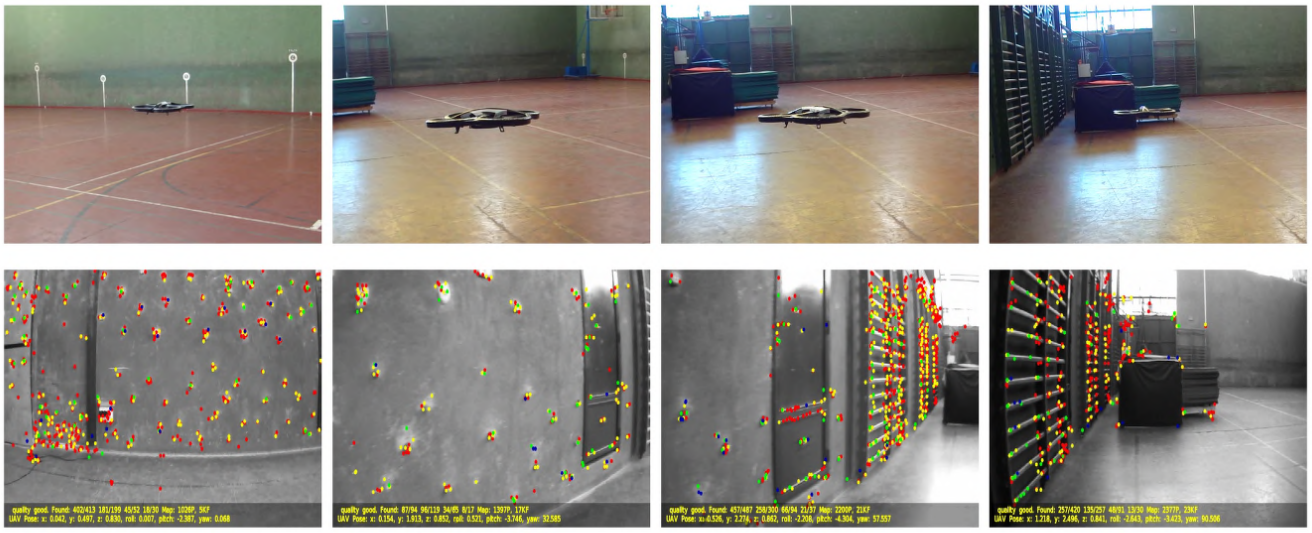


Fig. 12: See-and-Avoid task for UAV showing with external images and real-time processing images in rows, where, the first column: forward flight (Yaw Estimation: 0.068^o), the second column: avoiding with little turning (Yaw Estimation: 32.585^o), the third column: avoiding with big turning (Yaw Estimation: 57.557^o) and the last column: finish the see-and-avoid task (Yaw Estimation: 90.506^o).

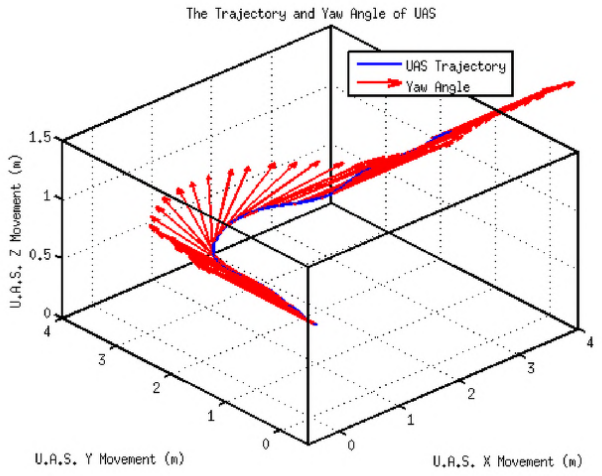


Fig. 13: 3D Recontruction for UAS Trajectory and Dynamic Change of Heading Angle using Pose Estimation Data. Where, along with Y Axis, NFA: 0-1m . SAA: 1-3m. DAA: 3-4m. Obstacle: 4m.

The related videos of these tests are shown in [26] and [29].

Figure 13 shows the 3D reconstruction of tracjectory and dynamic heading angle for UAS using the accurate pose estimation data.

VIII. Conclusion

This paper has presented a novel scaling factors and membership functions optimization-based fuzzy logic controller using Cross-Entropy technique. The training simulator is constructed based on the real see-and-avoid task in the Matlab Simulink. After the huge amount of simulations, two different optimized fuzzy controller were tested and compared with different pitch speeds and sizes of SAA. The new optimized fuzzy controller obtains excellent results against scaling factors optimization-based fuzzy. And this novel Cross-Entropy optimization not just improves the behavior of fuzzy controller, but also reduces 64% of the initial rule base.

For future works, as the field study [22] recommended, the omnidirectional sensor capabilities are needed for obstacle avoidance. For camera, the fisheye lense will be used. And a comparison between other different optimization techniques will be compared to this novel method. Finally, the optimization to rule weights of fuzzy controller will be done, which will casue Microscopic effects to the behavior of fuzzy controller.

Acknowledgment

The work reported in this paper is the consecution of several research stages at the Computer Vision Group - Universidad Politécnica de Madrid. This work has been sponsored by the Spanish Science and Technology Ministry under the grant CICYT DPI2010-20751-C02-01, ECHORD Project in the Eruropean FP7 and the China Scholarship Council (CSC).